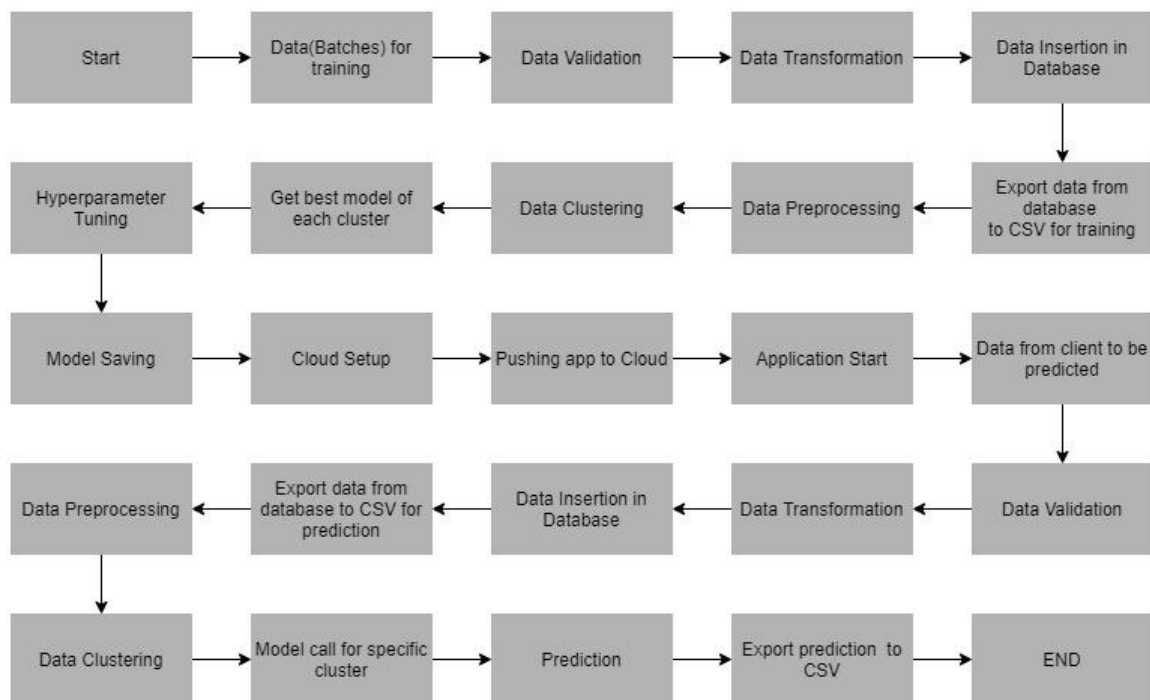

Problem Statement

To build a regression model to predict the concrete compressive strength based on the different features in the training data.

Architecture



Data Description

Given is the variable name, variable type, the measurement unit and a brief description.

The concrete compressive strength is the regression problem. The order of this listing

corresponds to the order of numerals along the rows of the database.

Name	Data Type	Measurement	Description
Cement (component 1)	quantitative	kg in a m3 mixture	Input Variable
Blast Furnace	quantitative	kg in a m3	Input Variable-- Blast furnace

Slag (component 2)		mixture	slag is a nonmetallic coproduct produced in the process. It consists primarily of silicates, aluminosilicates, and calcium-alumina-silicates
Fly Ash (component 3)	quantitative	kg in a m3 mixture	Input Variable- it is a coal combustion product that is composed of the particulates (fine particles of burned fuel) that are driven out of coal-fired boilers together with the flue gases.
Water (component 4)	quantitative	kg in a m3 mixture	Input Variable
Superplasticizer (component 5)	quantitative	kg in a m3 mixture	Input Variable-- Superplasticizers (SP's), also known as high range water reducers, are additives used in making high strength concrete. Their addition to concrete or mortar allows the reduction of the water to cement ratio without negatively affecting the workability of the mixture, and enables the production of self-consolidating concrete and high performance concrete
Coarse Aggregate (component 6)	quantitative	kg in a m3 mixture	Input Variable-- construction aggregate, or simply "aggregate", is a broad category of coarse to medium grained particulate material used in construction, including sand, gravel, crushed stone, slag, recycled concrete and geosynthetic aggregates

Fine Aggregate (component 7)	quantitative	kg in a m3 mixture	Input Variable—Similar to coarse aggregate, the constitution is much finer.
Age	quantitative	Day (1~365)	Input Variable
Concrete compressive strength	quantitative	MPa	Output Variable

Apart from training files, we also require a "schema" file from the client, which contains all the relevant information about the training files such as:

Name of the files, Length of Date value in FileName, Length of Time value in FileName, Number of Columns, Name of the Columns, and their datatype.

Data Validation

In this step, we perform different sets of validation on the given set of training files.

1. Name Validation- We validate the name of the files based on the given name in the schema file. We have created a regex pattern as per the name given in the schema file to use for validation. After validating the pattern in the name, we check for the length of date in the file name as well as the length of time in the file name. If all the values are as per requirement, we move such files to "Good_Data_Folder" else we move such files to "Bad_Data_Folder."
2. Number of Columns - We validate the number of columns present in the files, and if it doesn't match with the value given in the schema file, then the file is moved to "Bad_Data_Folder."
3. Name of Columns - The name of the columns is validated and should be the same as given in the schema file. If not, then the file is moved to "Bad_Data_Folder".

-
4. The datatype of columns - The datatype of columns is given in the schema file. This is validated when we insert the files into Database. If the datatype is wrong, then the file is moved to "Bad_Data_Folder".
 5. Null values in columns - If any of the columns in a file have all the values as NULL or missing, we discard such a file and move it to "Bad_Data_Folder".

Data Insertion in Database

- 1) Database Creation and connection - Create a database with the given name passed. If the database is already created, open the connection to the database.
- 2) Table creation in the database - Table with name - "Good_Data", is created in the database for inserting the files in the "Good_Data_Folder" based on given column names and datatype in the schema file. If the table is already present, then the new table is not created and new files are inserted in the already present table as we want training to be done on new as well as old training files.
- 3) Insertion of files in the table - All the files in the "Good_Data_Folder" are inserted in the above-created table. If any file has invalid data type in any of the columns, the file is not loaded in the table and is moved to "Bad_Data_Folder".

Model Training

- 1) Data Export from Db - The data in a stored database is exported as a CSV file to be used for model training.
- 2) Data Preprocessing
 - a) Check for null values in the columns. If present, impute the null values using the KNN imputer
 - b) transform the features using log transformation
 - c) Scale the training and test data separately

3) Clustering - KMeans algorithm is used to create clusters in the preprocessed data. The optimum number of clusters is selected by plotting the elbow plot, and for the dynamic selection of the number of clusters, we are using "KneeLocator" function. The idea behind clustering is to implement different algorithms

To train data in different clusters. The Kmeans model is trained over preprocessed data and the model is saved for further use in prediction.

4) Model Selection - After clusters are created, we find the best model for each cluster. We are using two algorithms, "Random forest Regressor" and "Linear Regression". For each cluster, both the algorithms are passed with the best parameters derived from GridSearch. We calculate the Rsquared scores for both models and select the model with the best score. Similarly, the model is selected for each cluster. All the models for every cluster are saved for use in prediction.

Prediction Data Description

Client will send the data in multiple set of files in batches at a given location. Data will contain climate indicators in 8 columns.

Apart from prediction files, we also require a "schema" file from client which contains all the relevant information about the training files such as:

Name of the files, Length of Date value in FileName, Length of Time value in FileName, Number of Columns, Name of the Columns and their datatype.

Data Validation

In this step, we perform different sets of validation on the given set of training files.

1) Name Validation- We validate the name of the files on the basis of given Name in the schema file. We have created a regex pattern as per the name given in schema file, to use for validation. After validating the pattern in the name, we check for length of date in the file name as well as length of time in the file name. If all the values are as per requirement, we move such files to "Good_Data_Folder" else we move such files to "Bad_Data_Folder".

-
- 2) Number of Columns - We validate the number of columns present in the files, if it doesn't match with the value given in the schema file then the file is moved to "Bad_Data_Folder".
 - 3) Name of Columns - The name of the columns is validated and should be same as given in the schema file. If not, then the file is moved to "Bad_Data_Folder".
 - 4) Datatype of columns - The datatype of columns is given in the schema file. This is validated when we insert the files into Database. If datatype is wrong then the file is moved to "Bad_Data_Folder".
 - 5) Null values in columns - If any of the columns in a file has all the values as NULL or missing, we discard such file and move it to "Bad_Data_Folder".

Data Insertion in Database

- 1) Database Creation and connection - Create database with the given name passed. If the database is already created, open the connection to the database.
- 2) Table creation in the database - Table with name - "Good_Data", is created in the database for inserting the files in the "Good_Data_Folder" on the basis of given column names and datatype in the schema file. If table is already present then new table is not created, and new files are inserted the already present table as we want training to be done on new as well old training files.
- 3) Insertion of files in the table - All the files in the "Good_Data_Folder" are inserted in the above-created table. If any file has invalid data type in any of the columns, the file is not loaded in the table and is moved to "Bad_Data_Folder".

Prediction

- 1) Data Export from Db - The data in the stored database is exported as a CSV file to be used for prediction.
- 2) Data Preprocessing
 - a) Check for null values in the columns. If present, impute the null values using the KNN imputer

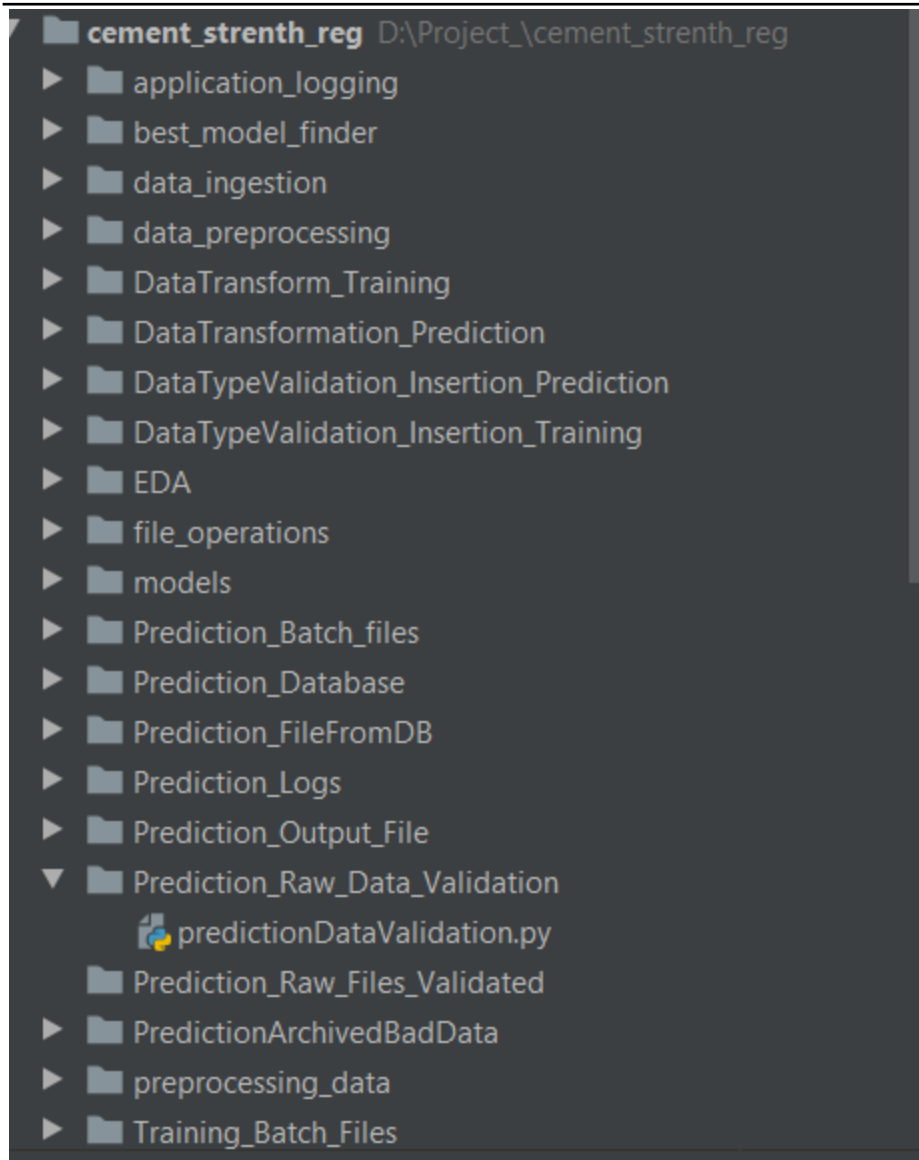
-
- b) transform the features using log transformation
 - c) Scale the training and test data separately
- 3) Clustering - KMeans model created during training is loaded, and clusters for the preprocessed prediction data is predicted.
 - 4) Prediction - Based on the cluster number, the respective model is loaded and is used to predict the data for that cluster.
 - 5) Once the prediction is made for all the clusters, the predictions along with the original names before label encoder are saved in a CSV file at a given location and the location is returned to the client.

Deployment

We will be deploying the model to the Pivotal Web Services Platform.

This is a workflow diagram for the prediction of using the trained model.

Now let's see the Cement_Strength project folder structure.



requirements.txt file consists of all the packages that you need to deploy the app in the cloud.


```

app.route("/predict", methods=['POST'])
cross_origin()
def predictRouteClient():
    try:
        if request.json['folderPath'] is not None:
            path = request.json['folderPath']

            pred_val = pred_validation(path) #object initialization

            pred_val.prediction_validation() #calling the prediction validation function

            pred = prediction(path) #object initialization

            # predicting for dataset present in database
            path = pred.predictionFromModel()
            return Response("Prediction File created at %s!!!" % path)

    except ValueError:
        return Response("Error Occurred! %s" %ValueError)
    except KeyError:
        return Response("Error Occurred! %s" %KeyError)
    except Exception as e:
        return Response("Error Occurred! %c" %e)

```

main.py is the entry point of our application, where the flask server starts.

```

def predictionFromModel(self):
    try:
        self.pred_data_val.deletePredictionFile() #deletes the existing prediction file from last run!
        self.log_writer.log(self.file_object,'Start of Prediction')
        data_getter=data_loader_prediction.Data_Getter_Pred(self.file_object,self.log_writer)
        data=data_getter.get_data()

        #code change
        # wafer_names=data['Wafer']
        # data=data.drop(labels=['Wafer'],axis=1)

        preprocessor=preprocessing.Preprocessor(self.file_object,self.log_writer)

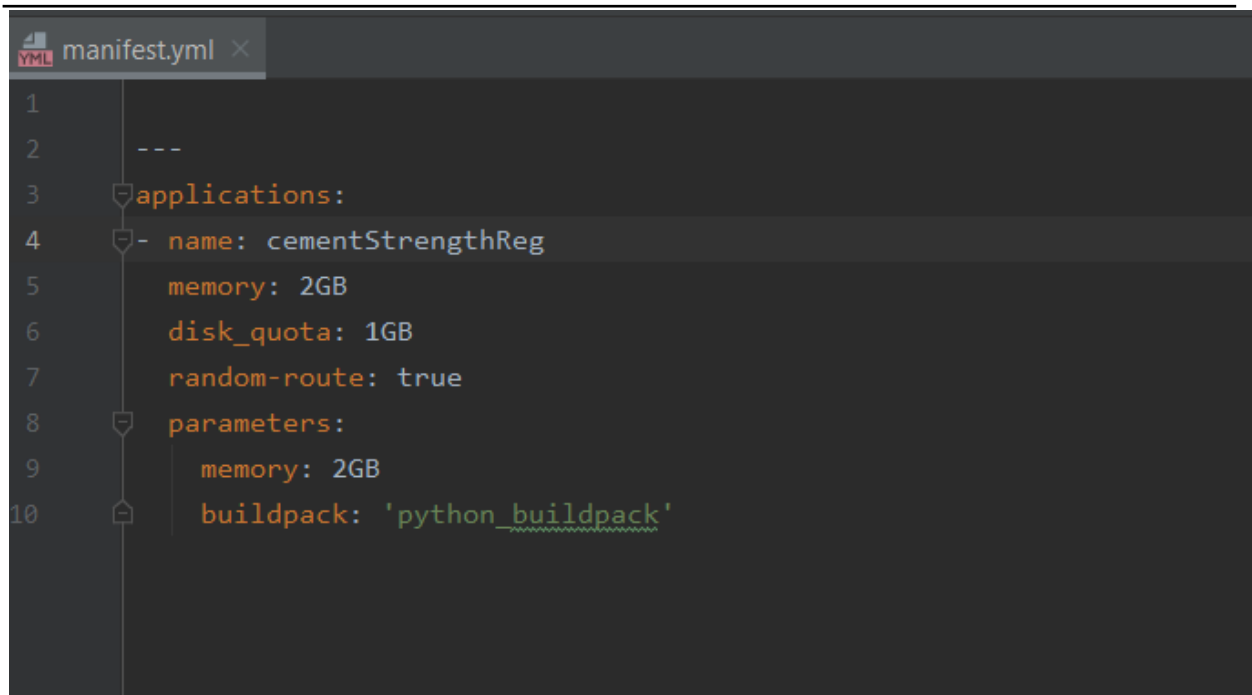
        is_null_present,cols_with_missing_values=preprocessor.is_null_present(data)
        if(is_null_present):
            data=preprocessor.impute_missing_values(data)

        data__ = preprocessor.logTransformation(data)

        #scale the prediction data
        data_scaled = pandas.DataFrame(preprocessor.standardScalingData(data),columns=data.columns)

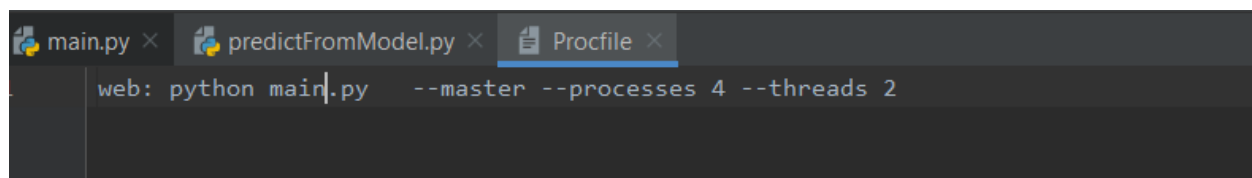
```

This is the **predictionFromModel.py** file where the predictions take place based on the data we are giving input to the model.



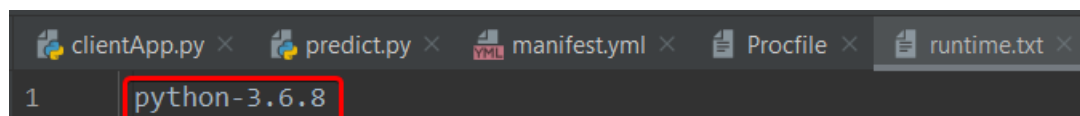
```
1 ---
2
3 applications:
4   - name: cementStrengthReg
5     memory: 2GB
6     disk_quota: 1GB
7     random-route: true
8     parameters:
9       memory: 2GB
10      buildpack: 'python_buildpack'
```

manifest.yml:- This file contains the instance configuration, app name, and build pack language.



```
web: python main.py --master --processes 4 --threads 2
```

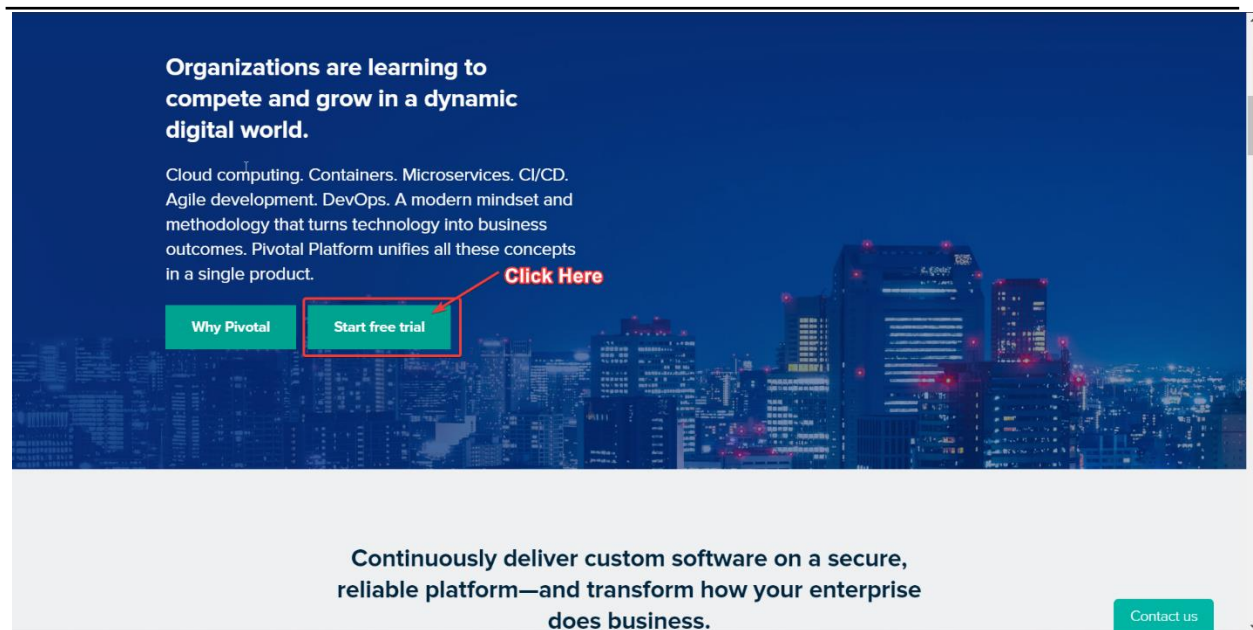
Procfile :- It contains the entry point of the app.



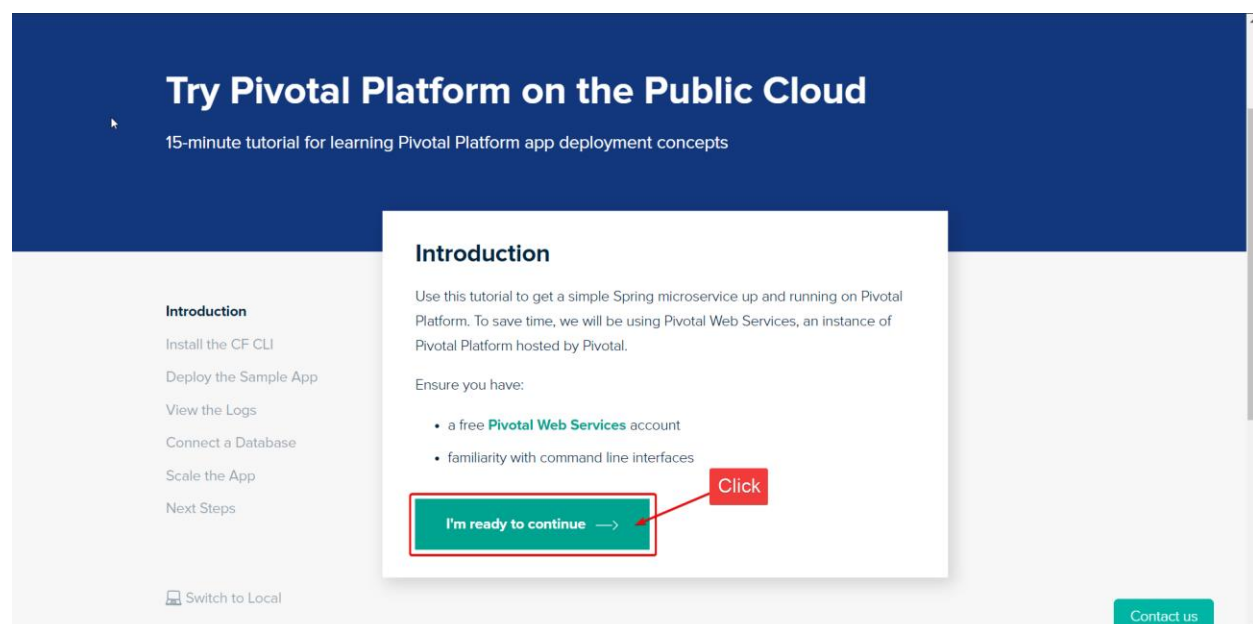
```
1 python-3.6.8
```

runtime.txt:- It contains the Python version number.

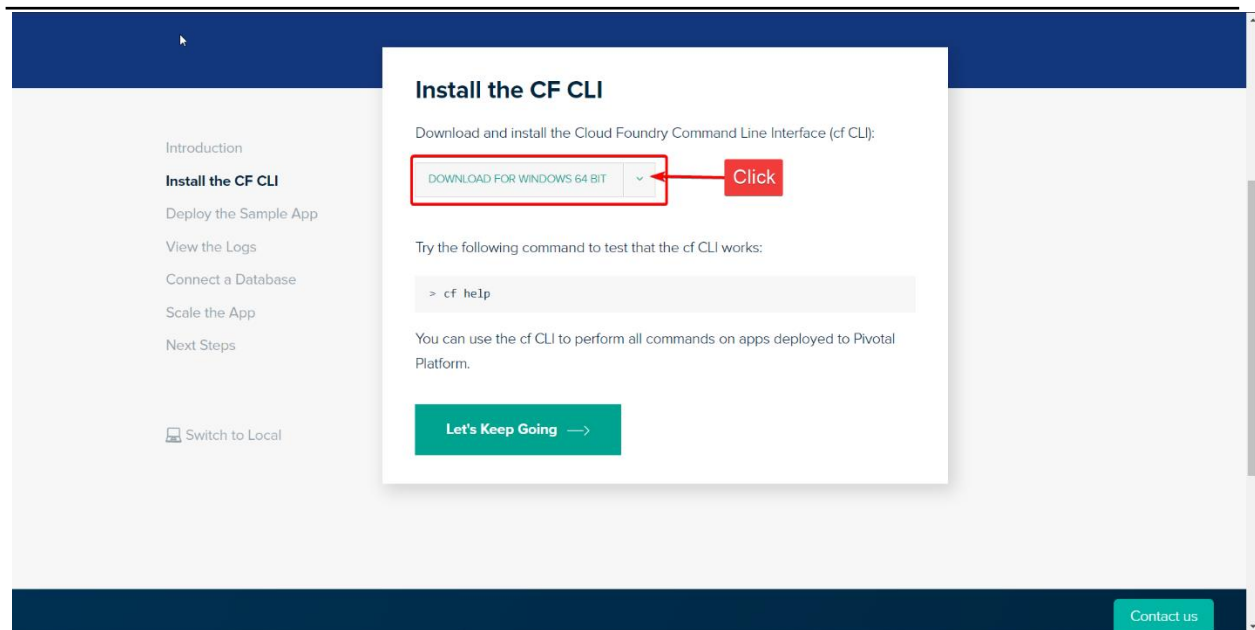
Visit the official website <https://pivotal.io/platform>.



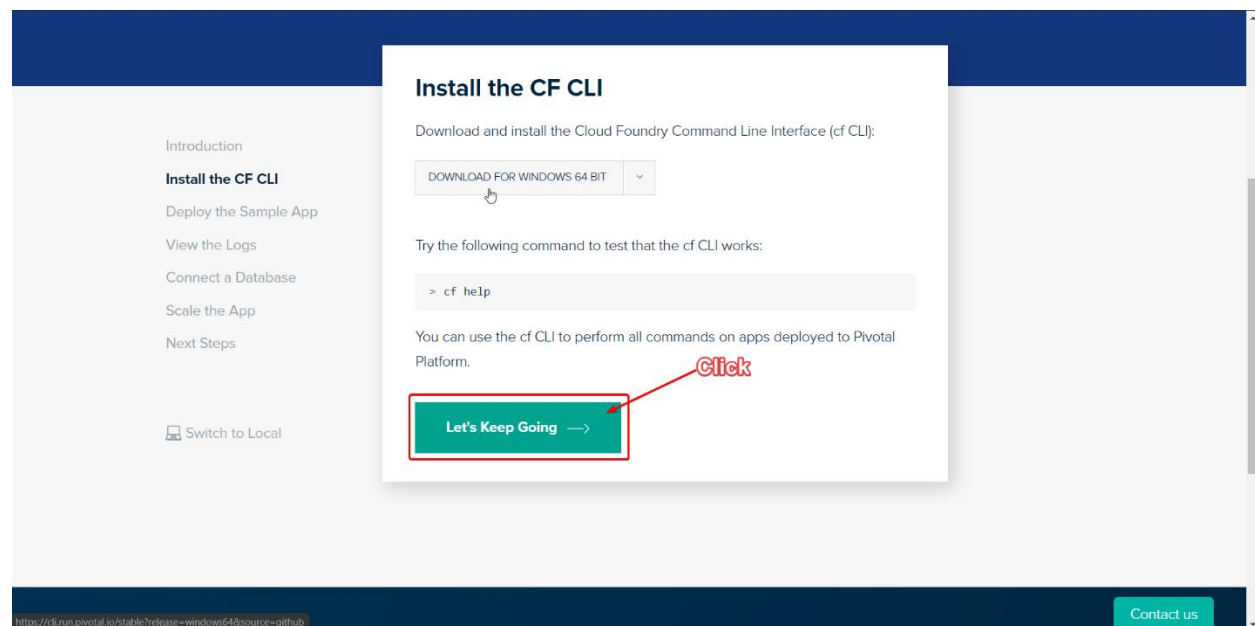
Scroll Down to see the **Start Trial Button**



Click on the start trial button and the next interface will open. Then I will click on **I'm ready to continue**



Click on Download for **Windows 64 bit**, and then zip file will be downloaded. Keep it for future uses.



Now click on **Let's Keep Going**



Sign in to continue

Next

Create account

Click

©2019 Pivotal Software, Inc. All Rights Reserved. [Privacy Policy](#) — [Terms of Use](#)

Click on **Create Your Account**



Create your Pivotal Account

First name

Last name

Email address

Password

Password confirmation

☐ I agree to the terms of Pivotal's [Privacy Policy](#)

Sign Up

Already have an account? [Sign In](#)

Fill Up your Details For registration

Do the email verification

Then login in again

Pivotal Web Services

sourangshuineuron@gmail.com

1 SIGN UP 2 CLAIM YOUR TRIAL 3 CREATE ORG

Sign Up for your free trial

Welcome to Pivotal Web Services! Complete these steps to access your account.

Username
sourangshuineuron@gmail.com

* Company
iNeuron

☒ I have read and agree to the Terms of Service for Pivotal Web Services

Next: Claim Your Trial

Pivotal © 2019 Pivotal Software Inc. All rights reserved. [Terms](#) [Privacy](#)

After logging you will see this screen below and start your free trial.

Write any Company or which one you prefer

Pivotal Web Services

sourangshuineuron@gmail.com

1 SIGN UP 2 CLAIM YOUR TRIAL 3 CREATE ORG

Claim Your Free Trial

Country *
United States

Mobile Number *

Send me my code

Already claimed your free trial? [Create a paid Org](#)

Your number is only used for claiming your free trial, and will never be distributed to third-parties or used for marketing purposes.

Users are limited to one free trial org per user account. If you have any issues or questions, please contact support@run.pivotal.io.

Pivotal © 2019 Pivotal Software Inc. All rights reserved. [Terms](#) [Privacy](#)

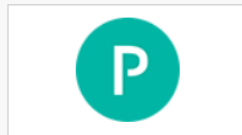
Enter your **Mobile Number** for Verification



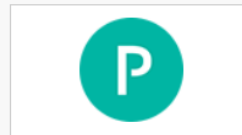
Where to?



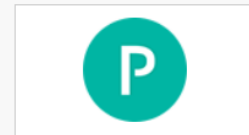
Pivotal Web Services



Pivotal Network



Partner Portal



Pivotal Support

Click on **Pivotal Web Services**

Pivotal Web Services

ORG

1 SIGN UP 2 CLAIM YOUR TRIAL 3 CREATE ORG

Create a New Org

Marketplace

Tools

Docs

Support

Blog

Status

Create a Trial Org

Org (or Project) Name *

App_Development

Have a promo code? Click here!

Start Free Trial

Organization (org) is a development account that encompasses computing resources, apps, and services. It can be owned and used by an individual or by multiple collaborators.

Set the org name to be the name of the project you'll be working on or the name of your team. Don't worry - you can change this name at any time!

Free Trial

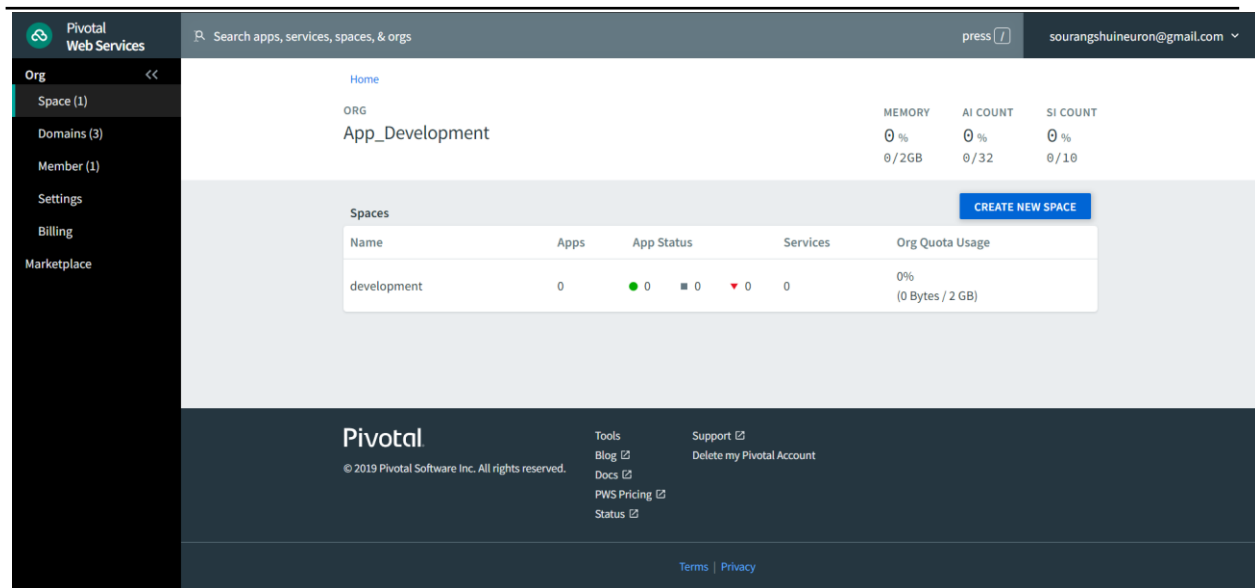
- \$87 credit for app usage to use for up to 1 year
- Up to 2GB of memory to share across app instances
- Choice of free marketplace services to try
- Unlimited collaborators

Upgrade at any time for

- Access to 25GB of memory at \$0.03/GB hr
- Premium service plans
- Pay for only what you use

Pivotal © 2019 Pivotal Software Inc. All rights reserved. [Terms](#) [Privacy](#)

Give any **Org** name



Now you are inside your Org, and by default, development space is created in your org. You can push your apps here.

The cloud signup process is done, and the setup is ready for us to push the app.

Previously you have downloaded the **CLI.zip** file. Unzip the file and install the .exe file with admin rights.

After a successful installation, you can verify by opening your CMD and type **cf**.

Then you will get a screen which is shown below

```

Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\soura>cf
cf version 6.46.1+4934877ec.2019-08-23, Cloud Foundry command line tool
Usage: cf [global options] command [arguments...] [command options]

Before getting started:
  config      login,l      target,t
  help,h      logout,lo

Application lifecycle:
  apps,a      run-task,rt   events
  push,p      logs          set-env,se
  start,st    ssh          create-app-manifest
  stop,sp     app          delete,d
  restart,rs  env,e
  restage,rg  scale

Services integration:
  marketplace,m      create-user-provided-service,cups
  services,s         update-user-provided-service,uups
  create-service,cs  create-service-key,csk
  update-service     delete-service-key,dsk
  delete-service,ds  service-keys,sk
  service            service-key
  bind-service,bs    bind-route-service,brs
  unbind-service,us  unbind-route-service,urs

Route and domain management:
  routes,r      delete-route   create-domain
  domains       map-route
  create-route  unmap-route

Space management:
  spaces      create-space   set-space-role
  space-users delete-space   unset-space-role

Org management:
  orgs,o      set-org-role
  org-users   unset-org-role

CLI plugin management:
  plugins      add-plugin-repo   repo-plugins
  install-plugin list-plugin-repos

Commands offered by installed plugins:

Global options:

```

If you see this screen in your CMD, the installation is successful.

Now type the command to login via cf-cli

```
cf login -a https://api.run.pivotal.io
```

Next, enter your email and password. Now you are ready to push your app.

Now let's go to the app which we have built.

```
Microsoft Windows [Version 10.0.18363.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\soura>cf login
API endpoint: https://api.run.pivotal.io

Email> sourangshuineuron@gmail.com

Password>
Authenticating...
OK

Targeted org App_Development

Targeted space development


API endpoint: https://api.run.pivotal.io (API version: 2.144.0)
User: sourangshuineuron@gmail.com
Org: App_Development
Space: development
```

Navigate to the project folder after downloading from the given below link:-

Then write `cf push` in the terminal.

```
D:\Project_\Thyroid_Detection>cf push
```

After the app is successfully deployed in the cloud, you will see the screen below with the route.

```
Cell 732975b5-a95c-4e37-b595-a0a3c3a9e2ea stopping instance 9f24f6ff-be7a-4eda-a6b3-81cb3c8bb315
Cell 732975b5-a95c-4e37-b595-a0a3c3a9e2ea destroying container for instance 9f24f6ff-be7a-4eda-a6b3-81cb3c8bb315
Cell 732975b5-a95c-4e37-b595-a0a3c3a9e2ea successfully destroyed container for instance 9f24f6ff-be7a-4eda-a6b3-81cb3c8bb315

Waiting for app to start...

name: WaferQuality
requested state: started
routes: waferquality-fearless-topi-vk.cfapps.io
last uploaded: Fri 31 Jan 21:01:17 IST 2020
stack: cflinuxfs3
buildpacks: python

type: web
instances: 1/1
memory usage: 2048M
start command: python main.py --master --processes 4 --threads 2
state since cpu memory disk details
#0 running 2020-01-31T15:32:10Z 20.1% 55.9M of 2G 800.8M of 2G

D:\Project_\WaferFaultDetection>
D:\Project_\WaferFaultDetection>
```

Finally, the app is pushed in the cloud.

Lets Open Postman and see the result.

